UCB-VI

Lucas Janson

CS/Stat 184(0): Introduction to Reinforcement Learning

Fall 2024

Today

- Feedback from last lecture
- Recap
- Warm-up: ExploreThenExploit for deterministic MDPs
- Why we don’t want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

Feedback from feedback forms

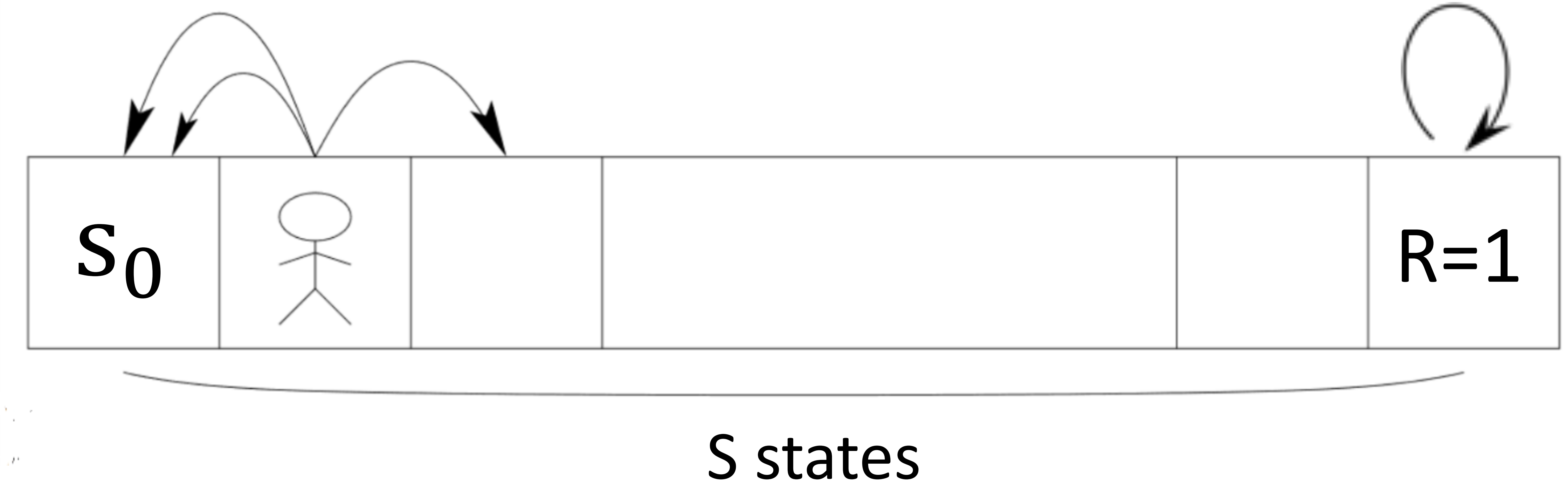
Feedback from feedback forms

1. Thank you to everyone who filled out the forms!

Today

- ✓ Feedback from last lecture
- Recap
- Warm-up: ExploreThenExploit for deterministic MDPs
- Why we don’t want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

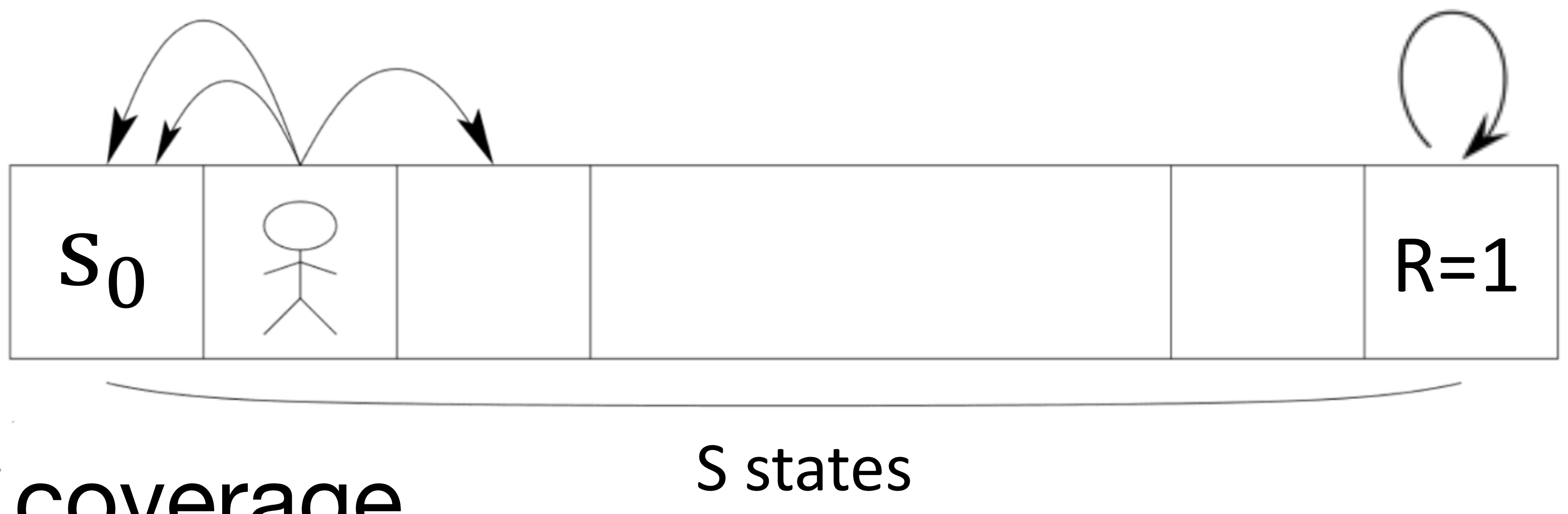
"Lack of Exploration" leads to Optimization and Statistical Challenges



Thrun '92

- Suppose $H \approx \text{poly}(|S|)$ & $\mu(s_0) = 1$ (i.e. we start at s_0).
- A randomly initialized policy π^0 has prob. $O(1/3^{|S|})$ of hitting the goal state in a trajectory.
- Thus a sample-based approach, with $\mu(s_0) = 1,$ require $O(3^{|S|})$ trajectories.
    - Holds for (sample based) Fitted DP
    - Holds for (sample based) PG/TRPO/NPG/PPO
- Basically, for these approaches, there is no hope of learning the optimal policy if $\mu(s_0) = 1.$

Let's examine the role of μ



Thrun '92

- Suppose that somehow the distribution μ had better coverage.
  - e.g, if μ was uniform overall states in our toy problem, then all approaches we covered would work (with mild assumptions)
  - Theory: TRPO/NPG/PPO have better guarantees than fitted DP methods (assuming some "coverage")
- Strategies without coverage:
  - If we have a simulator, sometimes we can design μ to have better coverage.
    - this is helpful for robustness as well.
  - Imitation learning (next time).
    - An expert gives us samples from a "good" μ.
  - Explicit exploration:
    - UCB-VI: we'll merge two good ideas!
    - Encourage exploration in PG methods.
  - Try with reward shaping

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

$$Q_{H-1}^\star(s,a) = r(s,a)$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H - 1$,

$$Q_{H-1}^\star(s,a) = r(s,a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

$$Q_{H-1}^\star(s,a) = r(s,a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

$$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a) = Q_{H-1}^\star(s, \pi_{H-1}^\star(s))$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H - 1$,

$$Q_{H-1}^\star(s, a) = r(s, a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s, a)$$

$$V_{H-1}^\star = \max_a Q_{H-1}^\star(s, a) = Q_{H-1}^\star(s, \pi_{H-1}^\star(s))$$

2. Assuming we have computed $V_{h+1}^\star$, $h \leq H - 2$, i.e., assuming we know how to perform optimally starting at $h + 1$, then:

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

$$Q_{H-1}^\star(s,a) = r(s,a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

$$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a) = Q_{H-1}^\star(s, \pi_{H-1}^\star(s))$$

2. Assuming we have computed $V_{h+1}^\star$, $h \leq H-2$, i.e., assuming we know how to perform optimally starting at $h+1$, then:

$$Q_h^\star(s,a) = r(s,a) + \mathbb{E}_{s' \sim P(s,a)} V_{h+1}^\star(s')$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

$$Q_{H-1}^\star(s,a) = r(s,a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

$$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a) = Q_{H-1}^\star(s, \pi_{H-1}^\star(s))$$

2. Assuming we have computed $V_{h+1}^\star$, $h \leq H-2$, i.e., assuming we know how to perform optimally starting at $h+1$, then:

$$Q_h^\star(s,a) = r(s,a) + \mathbb{E}_{s' \sim P(s,a)} V_{h+1}^\star(s')$$

$$\pi_h^\star(s) = \arg\max_a Q_h^\star(s,a),$$

Recall: Value Iteration (VI)

VI = DP is a backwards in time approach for computing the optimal policy:

$$\pi^\star = \{\pi_0^\star, \pi_1^\star, \ldots, \pi_{H-1}^\star\}$$

1. Start at $H-1$,

$$Q_{H-1}^\star(s,a) = r(s,a) \qquad \pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

$$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a) = Q_{H-1}^\star(s, \pi_{H-1}^\star(s))$$

2. Assuming we have computed $V_{h+1}^\star$, $h \le H-2$, i.e., assuming we know how to perform optimally starting at $h+1$, then:

$$Q_h^\star(s,a) = r(s,a) + \mathbb{E}_{s' \sim P(s,a)} V_{h+1}^\star(s')$$

$$\pi_h^\star(s) = \arg\max_a Q_h^\star(s,a), \qquad V_h^\star = \max_a Q_h^\star(s,a)$$

Recall: Upper Confidence Bound (UCB)

For $t = 0, \ldots, T-1$:

Choose the arm with the highest upper confidence bound, i.e.,

$$a_t = \arg\max_{k \in \{1,\ldots,K\}} \hat{\mu}_t^{(k)} + \sqrt{\ln(2TK/\delta)/2N_t^{(k)}}$$

Recall: Upper Confidence Bound (UCB)

For $t = 0, \ldots, T-1$:

Choose the arm with the highest upper confidence bound, i.e.,

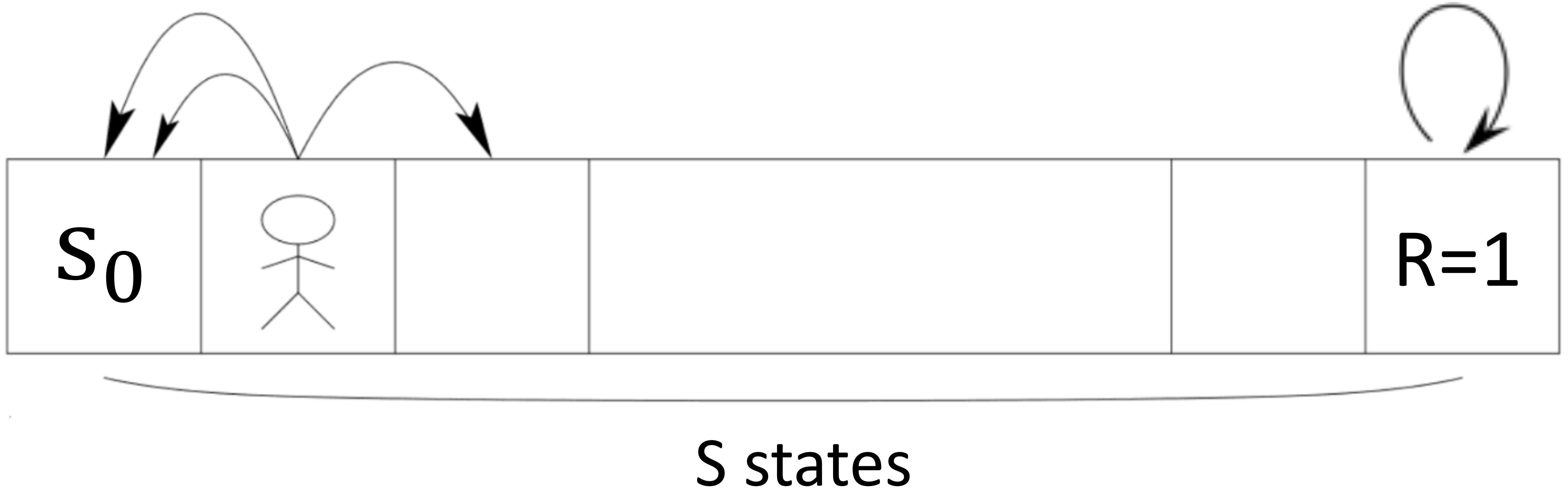
$$a_t = \arg\max_{k \in \{1,\ldots,K\}} \hat{\mu}_t^{(k)} + \sqrt{\ln(2TK/\delta)/2N_t^{(k)}}$$

High-level summary: estimate action quality, add exploration bonus, then argmax

Today

- Feedback from last lecture ✓
- Recap ✓
- Warm-up: ExploreThenExploit for deterministic MDPs
- Why we don't want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

How we do find $\pi^\star$ in an unknown MDP?

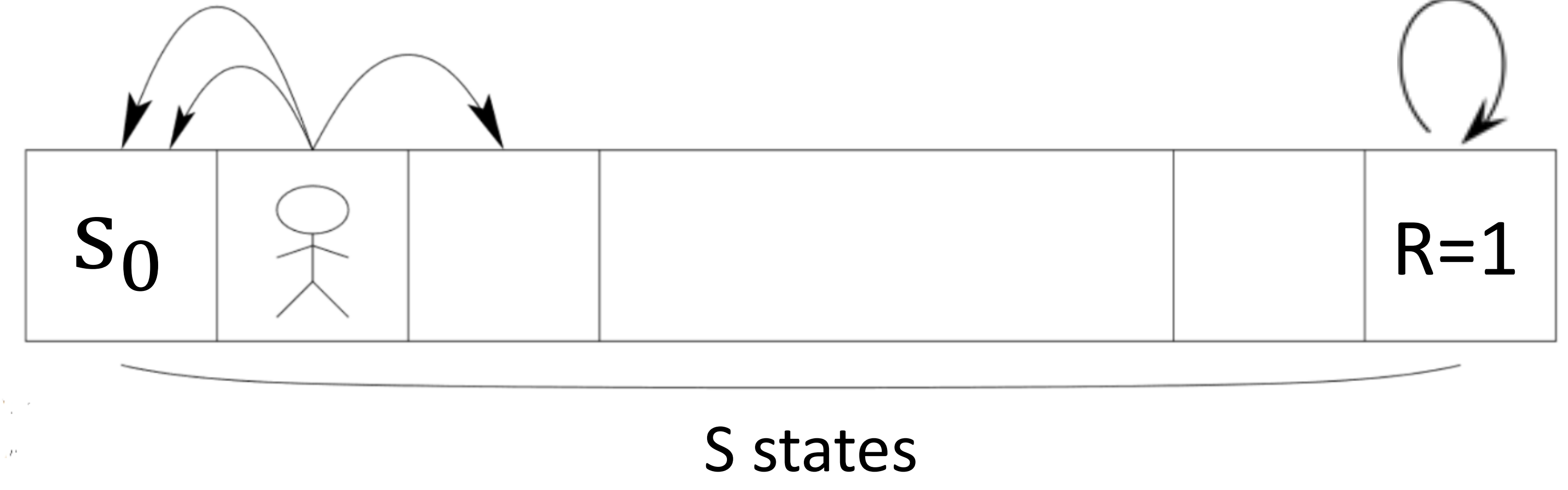


Thrun '92

- Episodic setting with an unknown MDP:
  - suppose we start at $s_0 \sim \mu$.
  - We act for H steps.
  - Then repeat.
- How do we find $\pi^\star$?
- How do we get low regret?

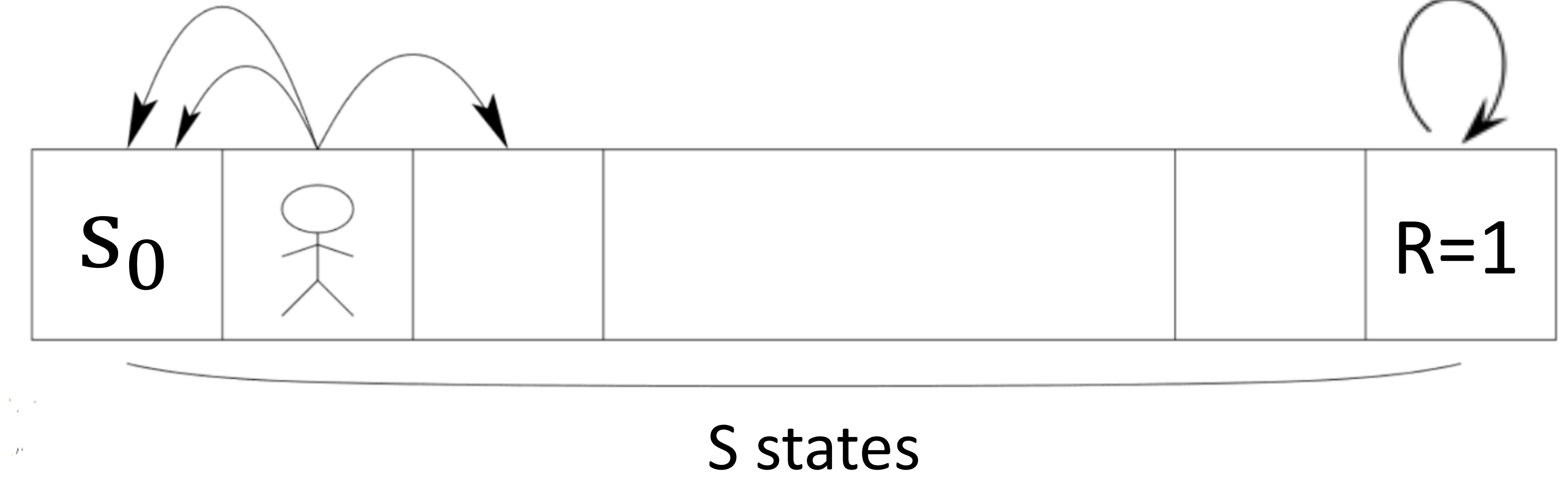
- Let's start with the setting where the MDP is deterministic.
  - So both $r(s, a)$ and $P(\cdot \mid s, a)$ are deterministic.

Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

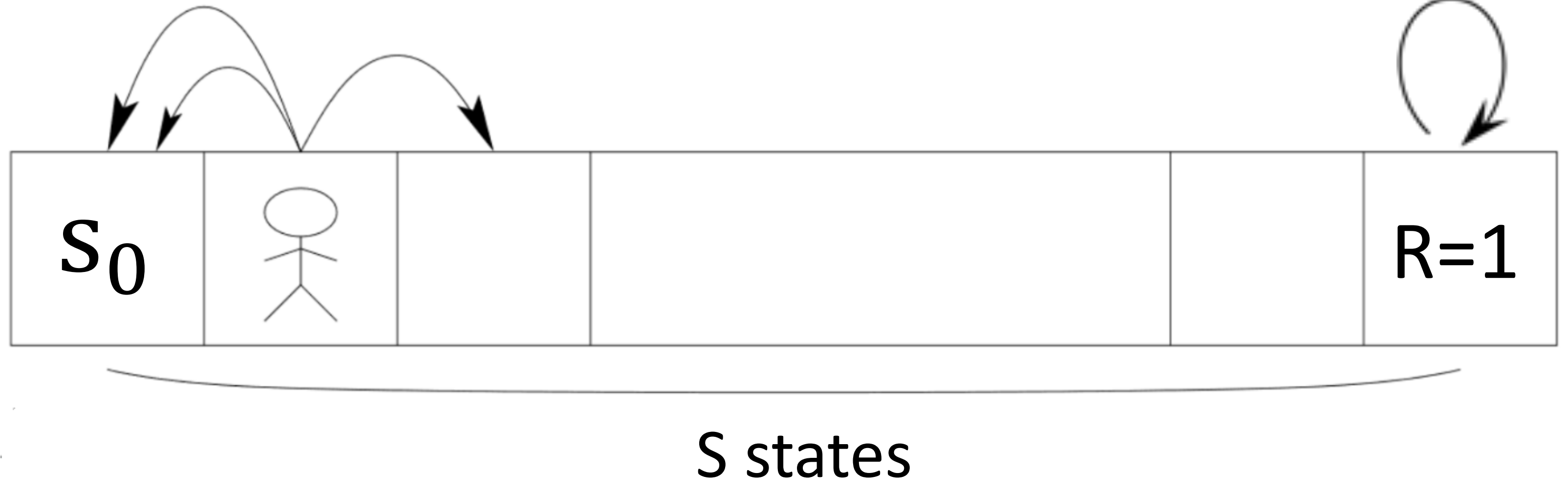
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.

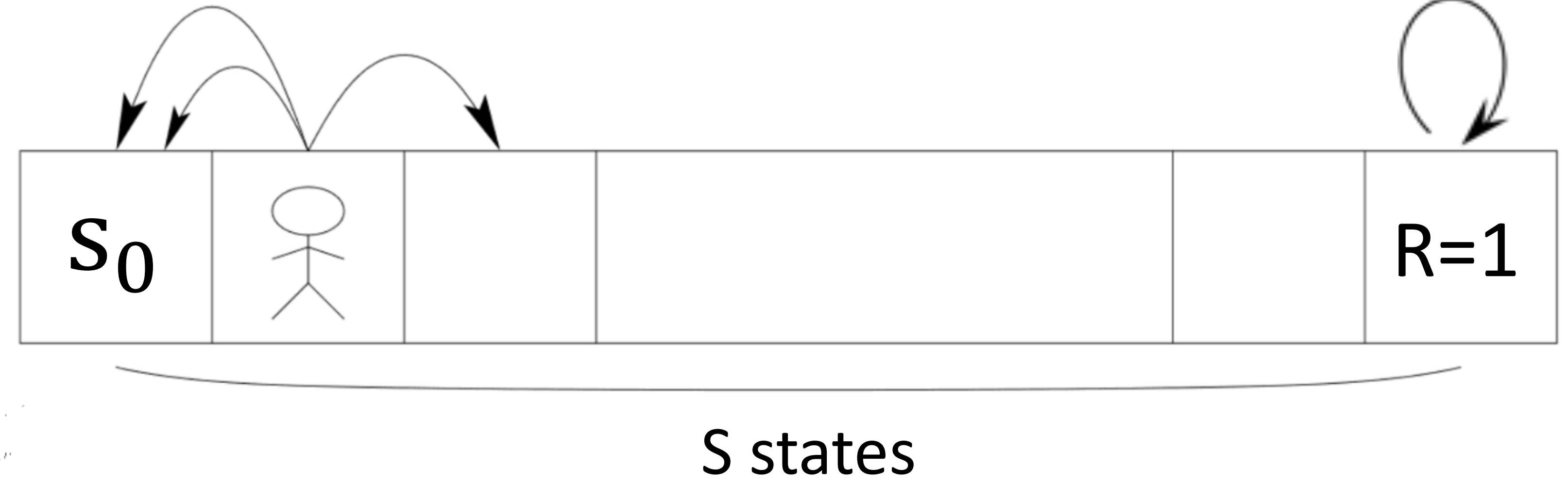
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?

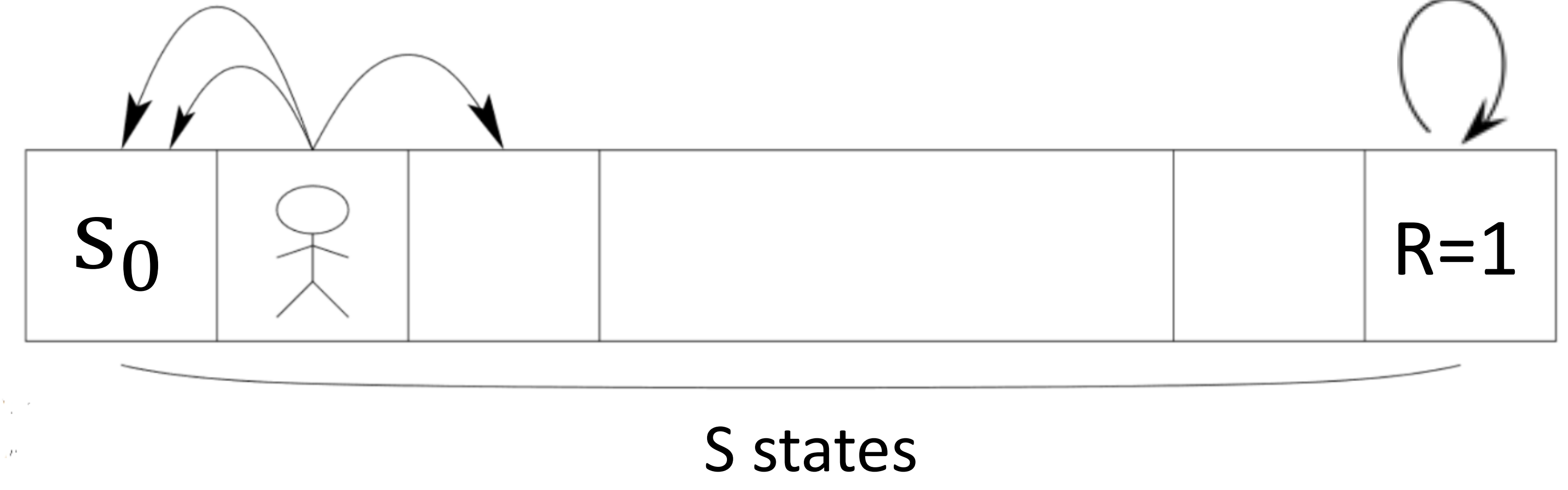
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes

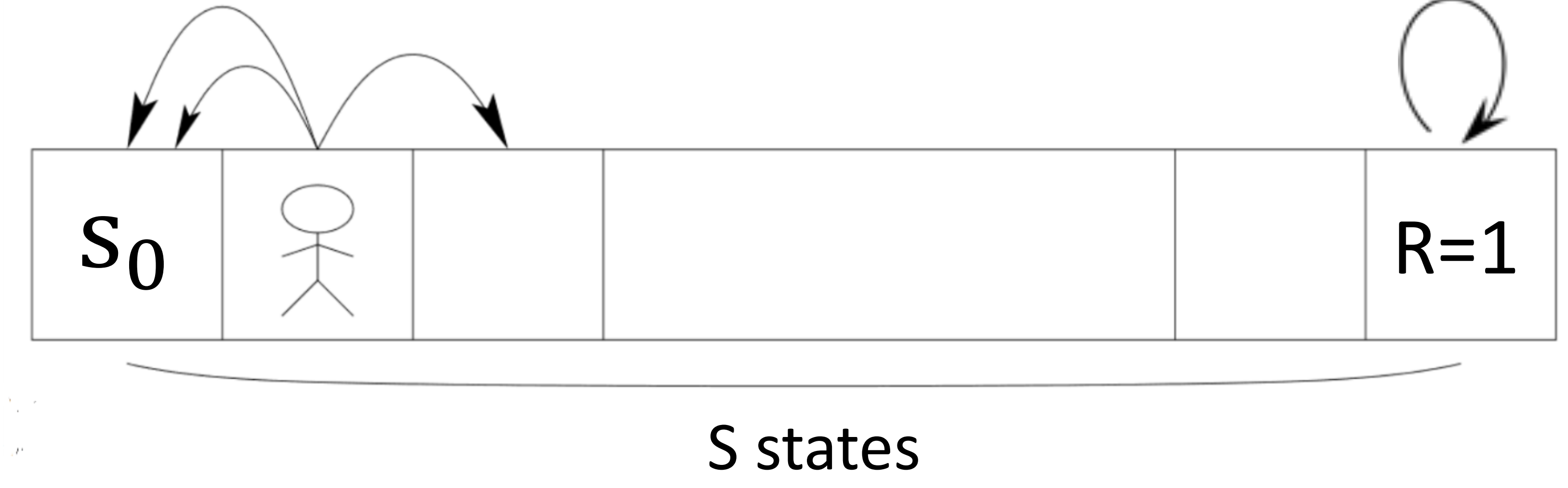
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:

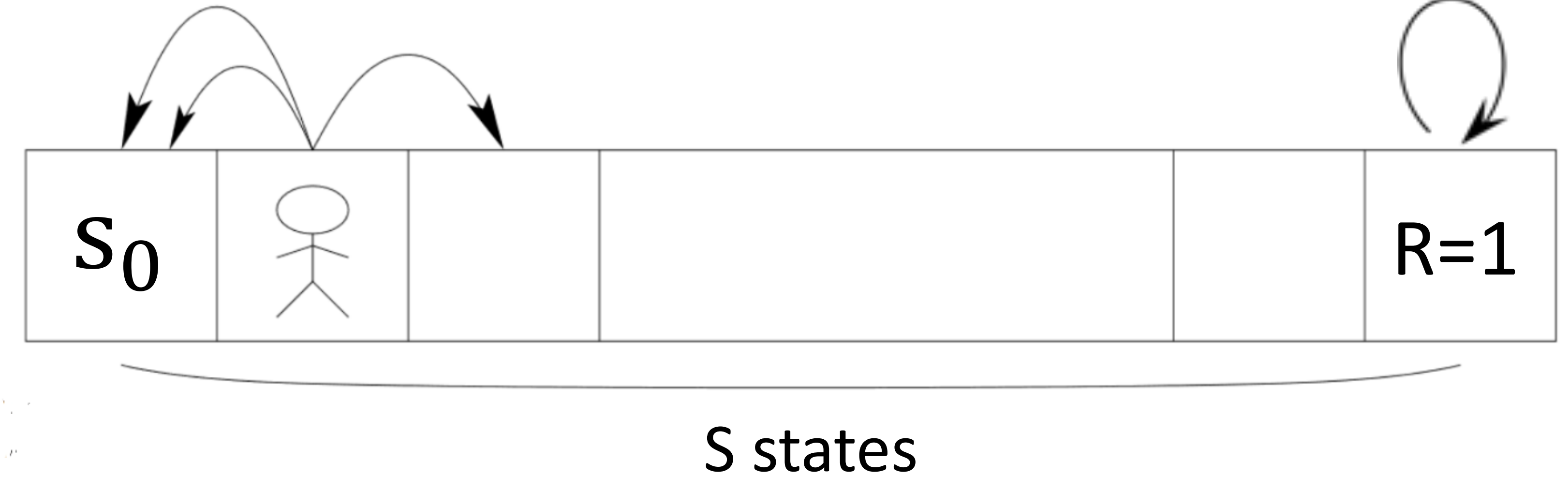
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
  - For $(s, a) \in K$,

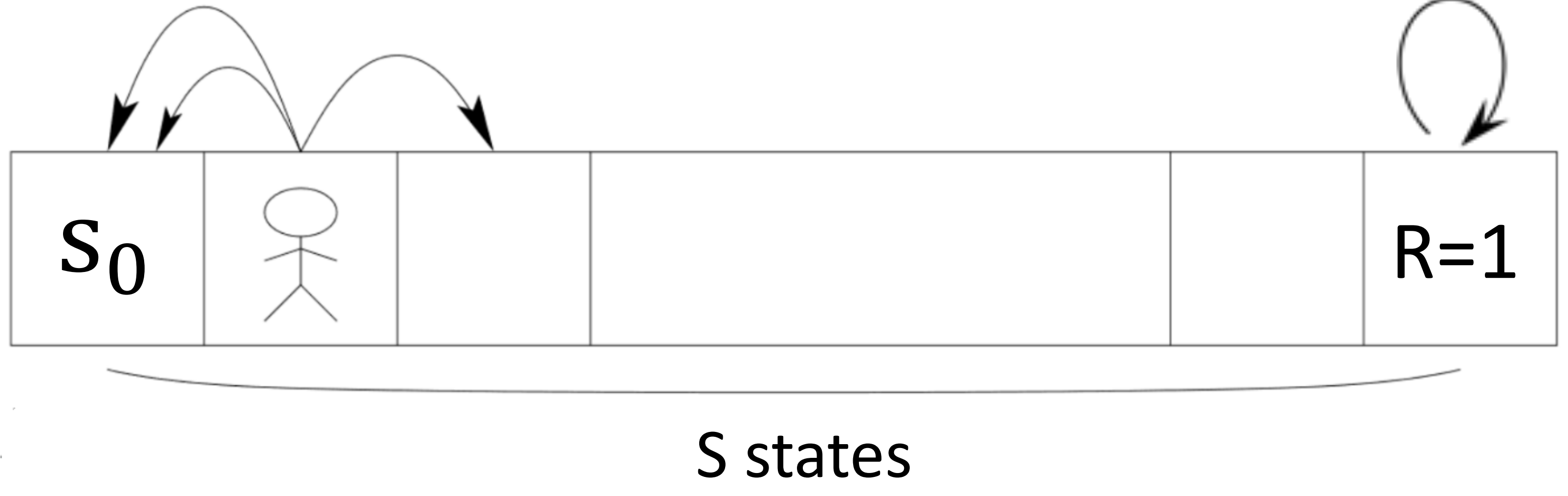
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
  - For $(s, a) \in K$,
    - define the dynamics in M_K to be same as in the true MDP.
      (note this is possible for us to do for $(s, a) \in K$)

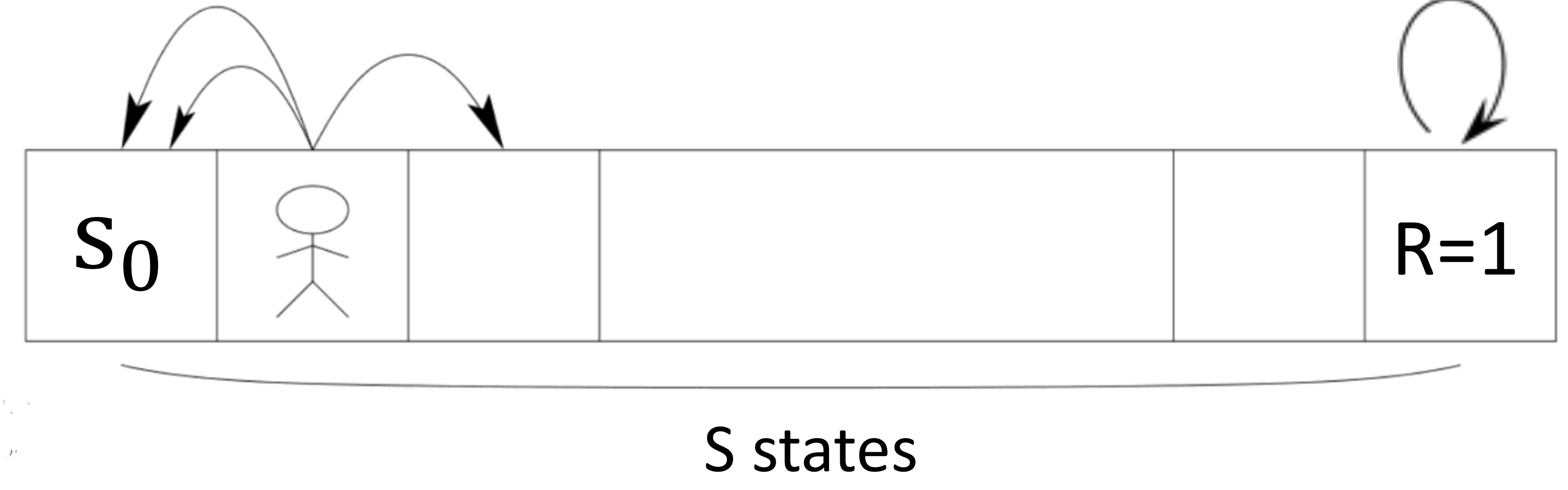
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
  - For $(s, a) \in K$,
    - define the dynamics in M_K to be same as in the true MDP. (note this is possible for us to do for $(s, a) \in K$)
    - define the reward as 0 for these state-action pairs.

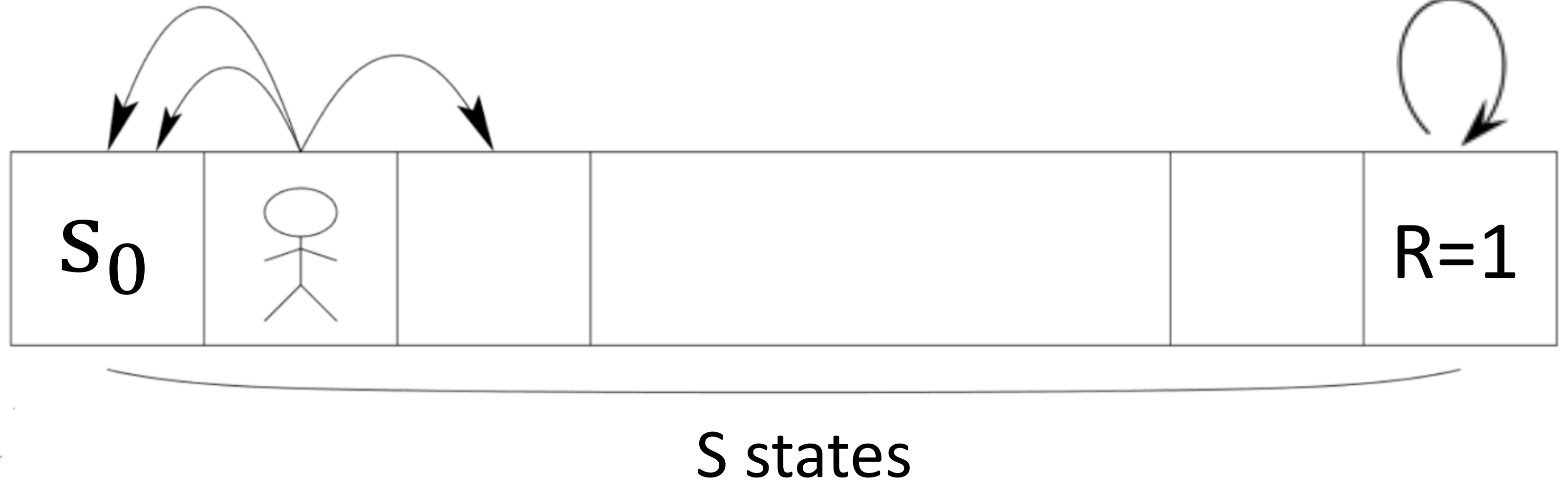
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
    - When is (s,a) known after a set of episodes?
    - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
    - For $(s, a) \in K$,
        - define the dynamics in M_K to be same as in the true MDP.
          (note this is possible for us to do for $(s, a) \in K$)
        - define the reward as 0 for these state-action pairs.
    - For $(s, a) \notin K$, assume we transition to a special state $s^\star$ which is absorbing (i.e., we stay at $s^\star$) and we always achieve a reward of 1 at this absorbing state.

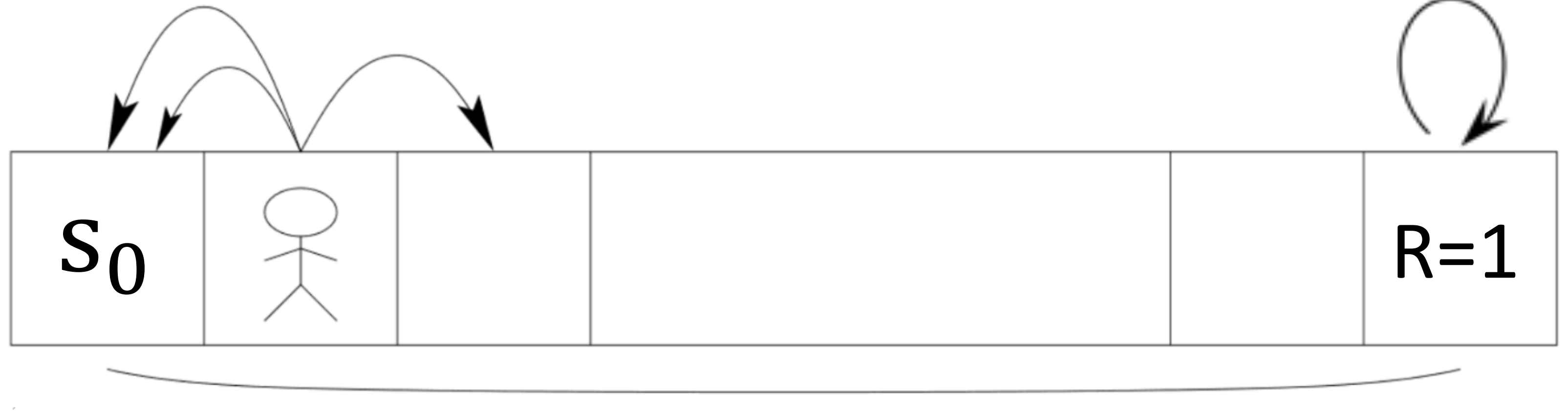
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
    - When is (s,a) known after a set of episodes?
    - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
    - For $(s, a) \in K$,
        - define the dynamics in M_K to be same as in the true MDP. (note this is possible for us to do for $(s, a) \in K$)
        - define the reward as 0 for these state-action pairs.
    - For $(s, a) \notin K$, assume we transition to a special state $s^\star$ which is absorbing (i.e., we stay at $s^\star$) and we always achieve a reward of 1 at this absorbing state.
- Let $\pi_K^\star$ and $V_K^\star$ be the optimal policy and value in M_K.

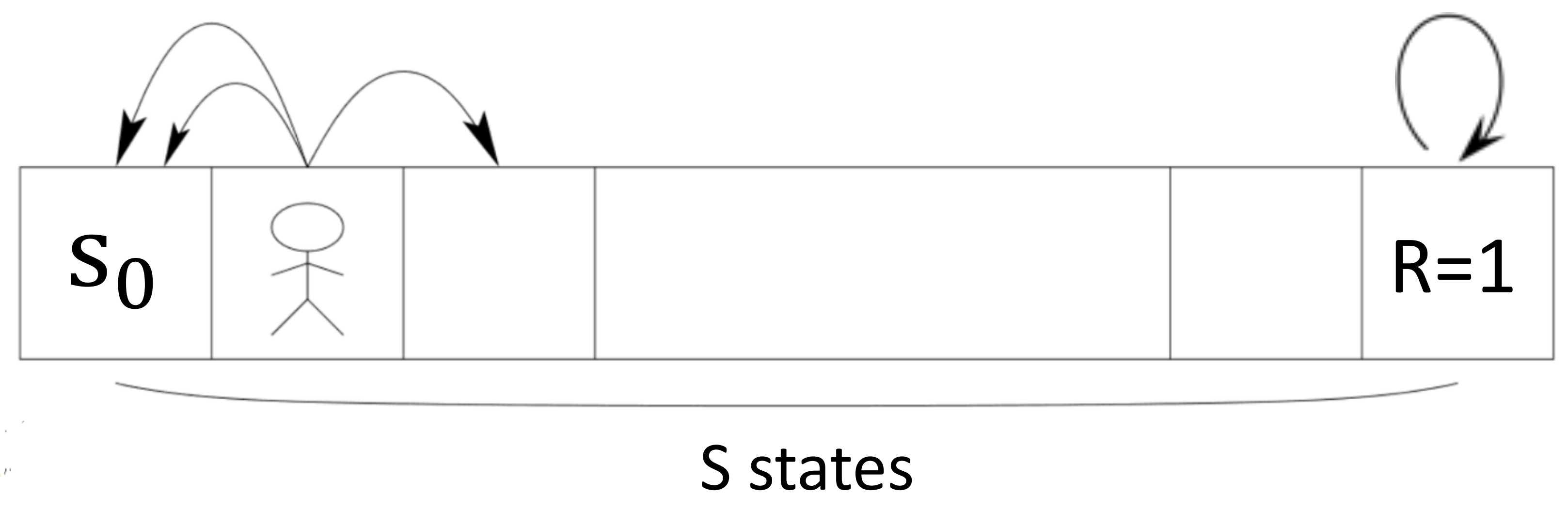
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

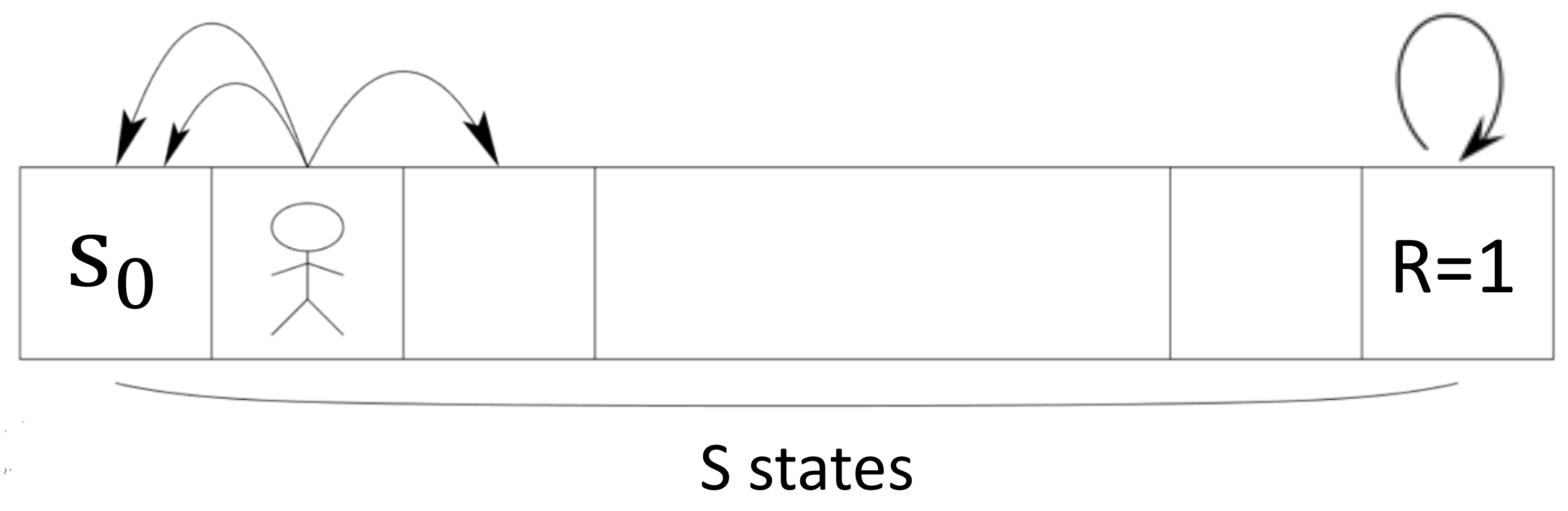
- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - When is (s,a) known after a set of episodes?
  - Let K be the set of known state-action pairs after a set of episodes
- Define the BonusMDP M_K with respect to the current (known) set K:
  - For $(s, a) \in K$,
    - define the dynamics in M_K to be same as in the true MDP. (note this is possible for us to do for $(s, a) \in K$)
    - define the reward as 0 for these state-action pairs.
  - For $(s, a) \notin K$, assume we transition to a special state $s^\star$ which is absorbing (i.e., we stay at $s^\star$) and we always achieve a reward of 1 at this absorbing state.
- Let $\pi_K^\star$ and $V_K^\star$ be the optimal policy and value in M_K.
- Theorem: Assume $H \geq |S|$.
  If K does not contain all state-action pairs, then $V_K^\star > 0$ and $\pi_K^\star$ will reach some $(s, a) \notin K$ (in at most $|S|$ steps).

Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

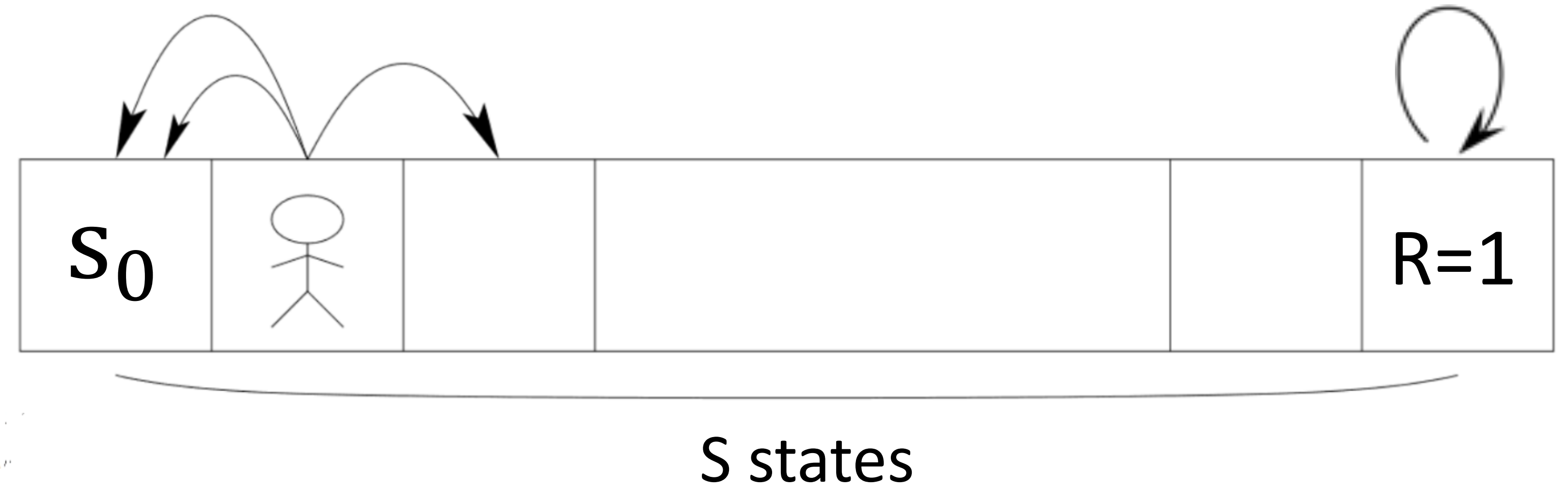
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.

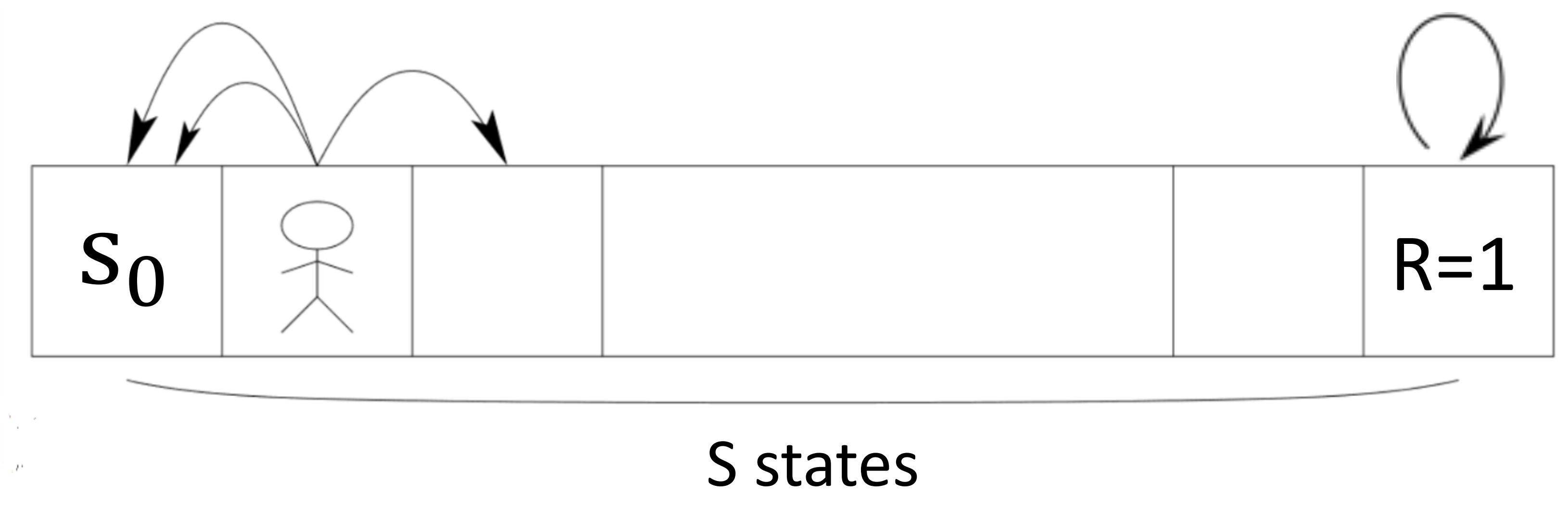
Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

Algorithm: ExploreThenExploit (for deterministic MDPs)

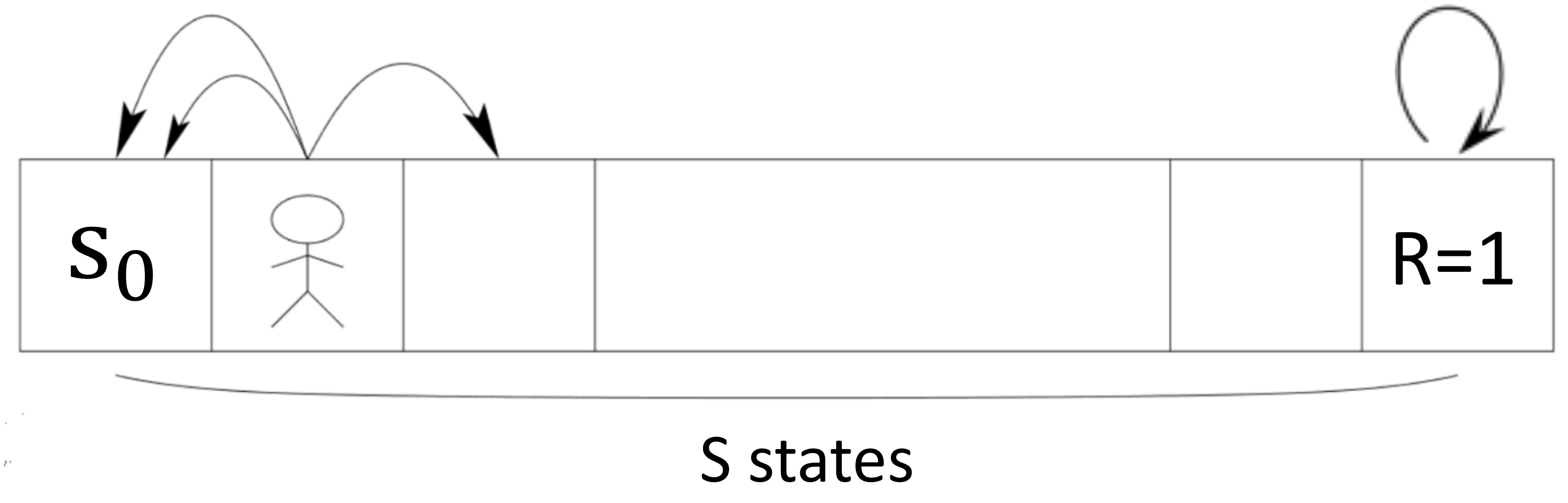


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \emptyset$

Algorithm: ExploreThenExploit (for deterministic MDPs)

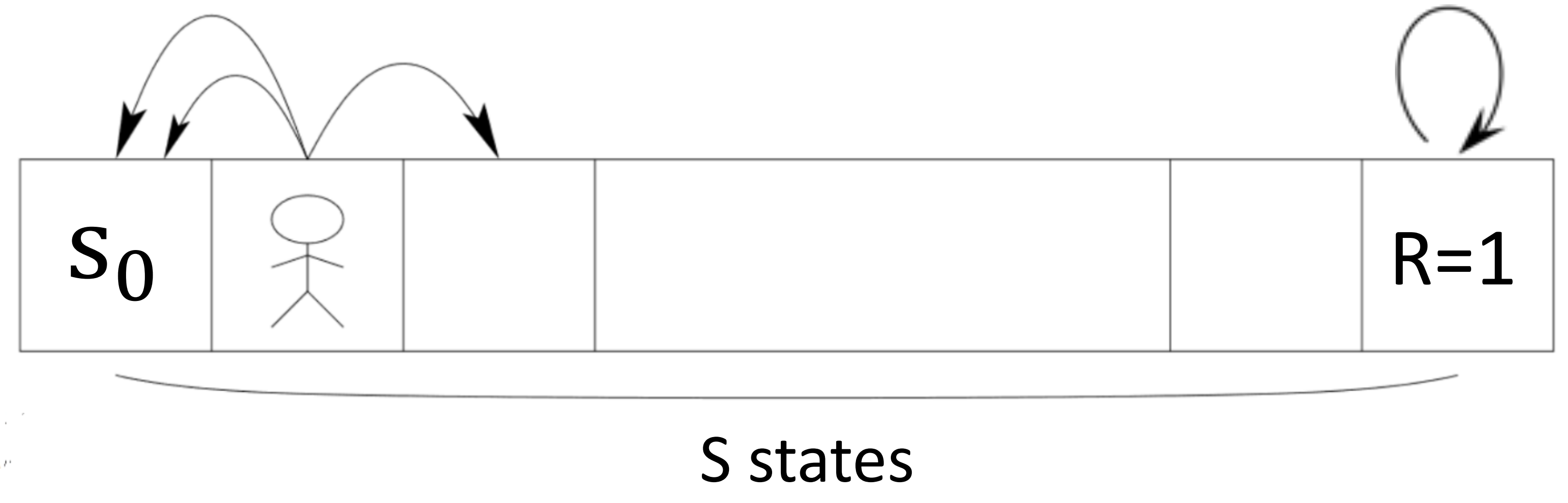


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \emptyset$
- While not terminated

Algorithm: ExploreThenExploit (for deterministic MDPs)

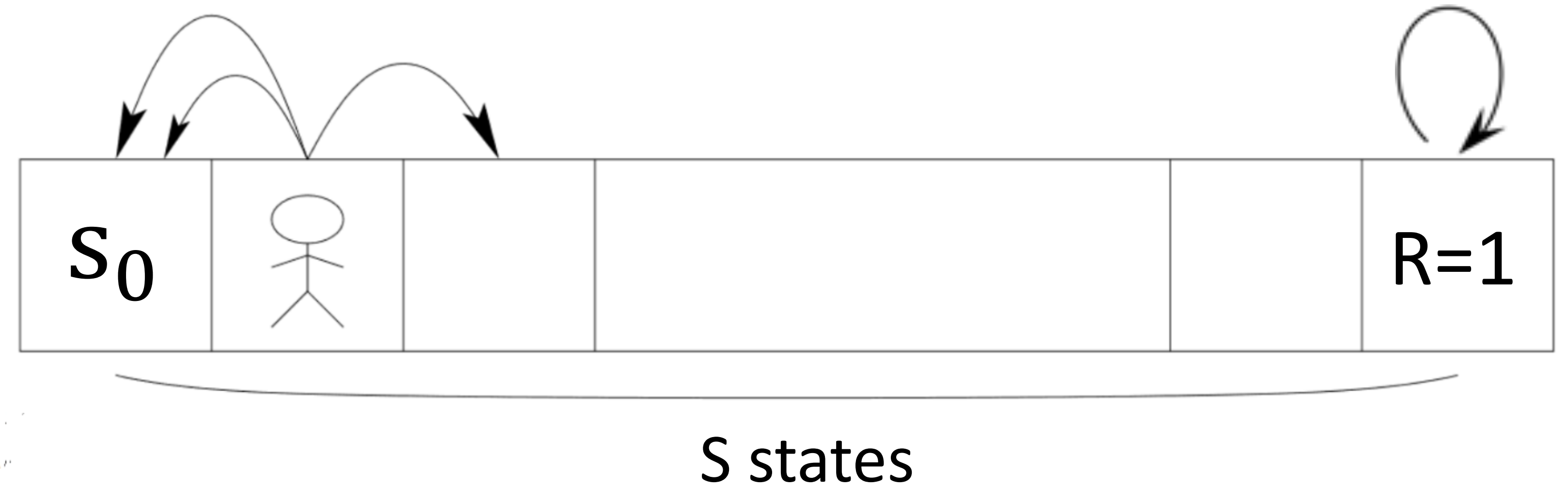


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \varnothing$
- While not terminated
  - Compute $\pi_K^\star$ and $V_K^\star$ for M_K.

Algorithm: ExploreThenExploit (for deterministic MDPs)

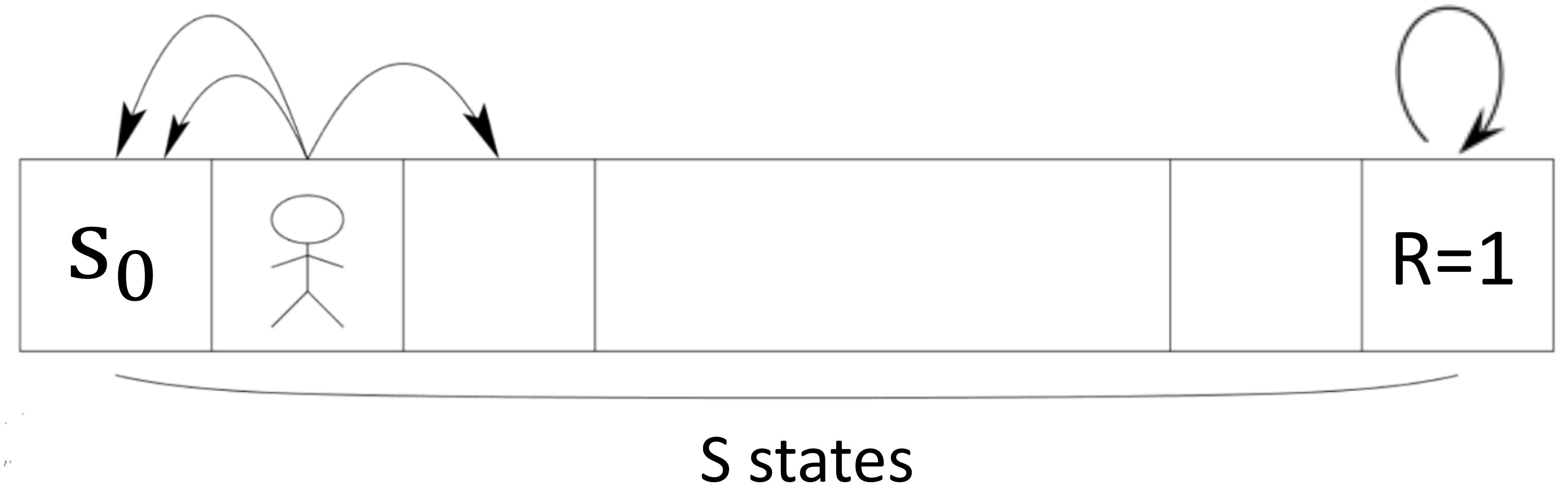


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \emptyset$
- While not terminated
  - Compute $\pi_K^\star$ and $V_K^\star$ for M_K.
    - If $V_K^\star > 0$, execute $\pi_K^\star$ and update the known set K

Algorithm: ExploreThenExploit (for deterministic MDPs)

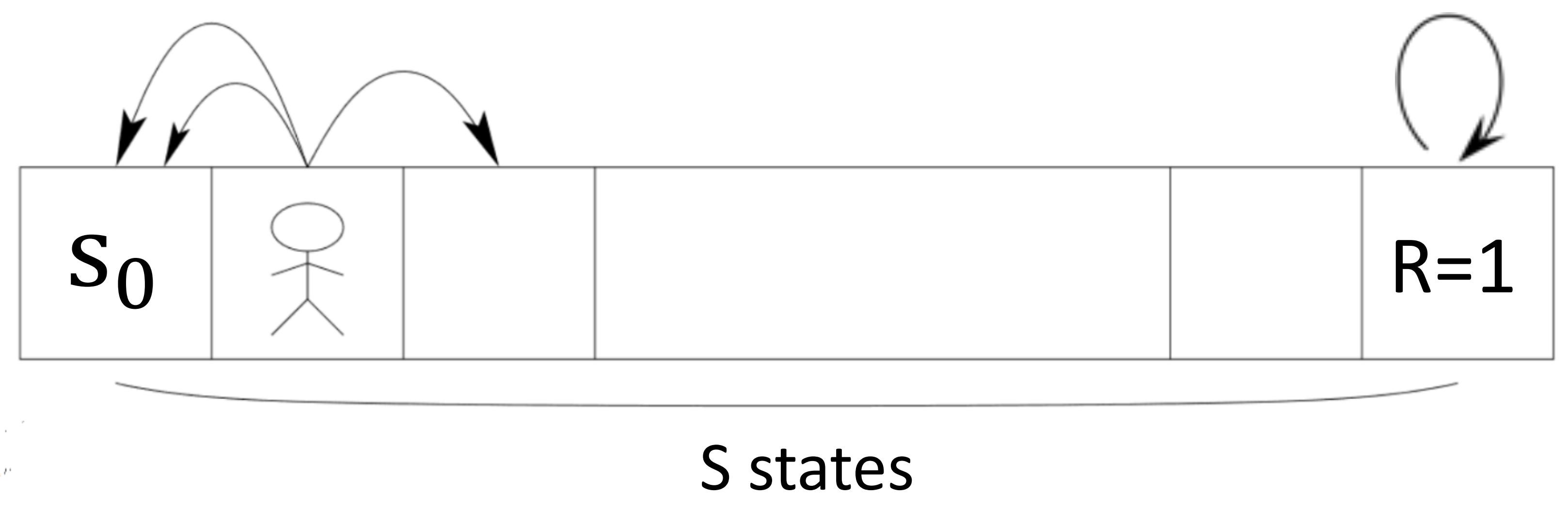


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \varnothing$
- While not terminated
  - Compute $\pi_K^\star$ and $V_K^\star$ for M_K.
    - If $V_K^\star > 0$, execute $\pi_K^\star$ and update the known set K
    - Else: terminate

Algorithm: ExploreThenExploit (for deterministic MDPs)

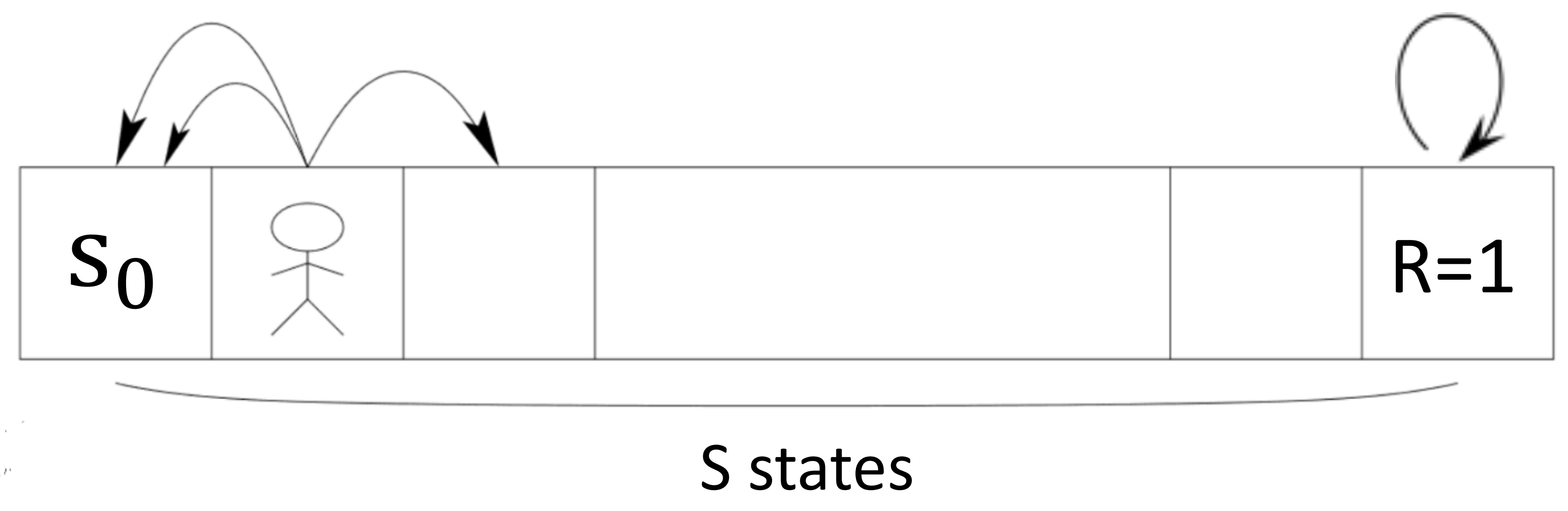


Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
  - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \varnothing$
- While not terminated
  - Compute $\pi^\star_K$ and $V^\star_K$ for M_K.
    - If $V^\star_K > 0$, execute $\pi^\star_K$ and update the known set K
    - Else: terminate
- Return: the optimal policy in the known MDP.

Algorithm: ExploreThenExploit (for deterministic MDPs)



Thrun '92

- Let's say a state-action pair (s,a) is known if both NextState(s, a) and $r(s, a)$ are known.
    - Let K be the set of known state-action pairs after a set of episodes

- Init: $K = \emptyset$
- While not terminated
    - Compute $\pi_K^\star$ and $V_K^\star$ for M_K.
        - If $V_K^\star > 0$, execute $\pi_K^\star$ and update the known set K
        - Else: terminate
- Return: the optimal policy in the known MDP.

Theorem: Assuming $H \geq |S|$, this algorithm returns an optimal policy in most $|S| \cdot |A|$ trajectories.

Comments:

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP

- How do we modify the algorithm for general H?

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP

- How do we modify the algorithm for general H?
  - Ignore any states that can't be reached in at most H steps!

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP
- How do we modify the algorithm for general H?
  - Ignore any states that can't be reached in at most H steps!
- What is the regret of this algorithm?

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP
- How do we modify the algorithm for general H?
    - Ignore any states that can't be reached in at most H steps!
- What is the regret of this algorithm?
    - Can be arbitrarily bad while searching, and searches for $|S||A|$ steps: $|S||A|H$

Comments:

- Basically formulating shortest path as an optimal policy in some modified MDP
- How do we modify the algorithm for general H?
  - Ignore any states that can't be reached in at most H steps!
- What is the regret of this algorithm?
  - Can be arbitrarily bad while searching, and searches for $|S||A|$ steps: $|S||A|H$
- Really needed determinism; for non-deterministic MDPs, need to think more like bandits…

Today

- ✓ Feedback from last lecture
- ✓ Recap
- ✓ Warm-up: ExploreThenExploit for deterministic MDPs
- Why we don’t want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

Exploration in MDP: make it a bandit and do UCB?

Q: given a discrete MDP, how many unique deterministic policies are there?

Exploration in MDP: make it a bandit and do UCB?

Q: given a discrete MDP, how many unique deterministic policies are there?

$$\left(|A|^{|S|} \right)^{H}$$

Exploration in MDP: make it a bandit and do UCB?

Q: given a discrete MDP, how many unique deterministic policies are there?

$$\left(|A|^{|S|} \right)^{H}$$

So treating each policy as an "arm" and running UCB gives us regret $\tilde{O}(\sqrt{|A|^{|S|H} N})$

Exploration in MDP: make it a bandit and do UCB?

Q: given a discrete MDP, how many unique deterministic policies are there?

$$\left(|A|^{|S|}\right)^H$$

So treating each policy as an "arm" and running UCB gives us regret $\tilde{O}(\sqrt{|A|^{|S|H}N})$

This seems bad, so are MDPs just super hard or can we do better?

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

All state transitions happen with probability 1/2 for all actions

Reward function: $$\begin{aligned} r(a,1) = r(b,1) = 0 \\ r(a,2) = r(b,2) = 1 \end{aligned}$$

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

All state transitions happen with probability 1/2 for all actions

Reward function: $$\begin{aligned} r(a,1) = r(b,1) = 0 \\ r(a,2) = r(b,2) = 1 \end{aligned}$$

Suppose we have a lot of data already on a policy $\pi^{(1)}$ that always takes action 1 and a policy $\pi^{(2)}$ that always takes action 2 (note $\pi^{(2)} = \pi^\star$)

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

All state transitions happen with probability 1/2 for all actions

Reward function: $$r(a,1) = r(b,1) = 0$$
$$r(a,2) = r(b,2) = 1$$

Suppose we have a lot of data already on a policy $\pi^{(1)}$ that always takes action 1 and a policy $\pi^{(2)}$ that always takes action 2 (note $\pi^{(2)} = \pi^{\star}$)

What do we know about a policy $\pi^{(3)}$ which always takes action 1 in the first time step, and always takes action 2 at the second time step?

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

All state transitions happen with probability 1/2 for all actions

Reward function: $$r(a,1) = r(b,1) = 0$$ $$r(a,2) = r(b,2) = 1$$

Suppose we have a lot of data already on a policy $\pi^{(1)}$ that always takes action 1 and a policy $\pi^{(2)}$ that always takes action 2 (note $\pi^{(2)} = \pi^\star$)

What do we know about a policy $\pi^{(3)}$ which always takes action 1 in the first time step, and always takes action 2 at the second time step?

Everything: we have a lot of data on every state-action reward and transition!

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

All state transitions happen with probability 1/2 for all actions

Reward function: $r(a,1) = r(b,1) = 0$
$r(a,2) = r(b,2) = 1$

Suppose we have a lot of data already on a policy $\pi^{(1)}$ that always takes action 1 and a policy $\pi^{(2)}$ that always takes action 2 (note $\pi^{(2)} = \pi^{\star}$)

What do we know about a policy $\pi^{(3)}$ which always takes action 1 in the first time step, and always takes action 2 at the second time step?

Everything: we have a lot of data on every state-action reward and transition!

If we treat the MDP as a bandit, we treat $\pi^{(3)}$ as a new "arm" about which we know nothing…

An example of MDP as bandit

$$S = \{a, b\}, \quad A = \{1,2\}, \quad H = 2$$

$|A|^{|S|H} = 2^4 = 16$

All state transitions happen with probability 1/2 for all actions

Reward function: $r(a,1) = r(b,1) = 0$
$r(a,2) = r(b,2) = 1$

Suppose we have a lot of data already on a policy $\pi^{(1)}$ that always takes action 1 and a policy $\pi^{(2)}$ that always takes action 2 (note $\pi^{(2)} = \pi^\star$)

What do we know about a policy $\pi^{(3)}$ which always takes action 1 in the first time step, and always takes action 2 at the second time step?

Everything: we have a lot of data on every state-action reward and transition!

If we treat the MDP as a bandit, we treat $\pi^{(3)}$ as a new "arm" about which we know nothing…

Today

- ✓ Feedback from last lecture
- ✓ Recap
- ✓ Warm-up: ExploreThenExploit for deterministic MDPs
- ✓ Why we don’t want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

UCB-VI: Tabular optimism in the face of uncertainty

Assume reward function $r_h(s, a)$ known

Inside iteration n :

UCB-VI: Tabular optimism in the face of uncertainty

Assume reward function $r_h(s, a)$ known

Inside iteration n :

Use all previous data to estimate dynamics $\{\hat{P}_h^n\}_{h=0}^{H-1}$

UCB-VI: Tabular optimism in the face of uncertainty

Assume reward function $r_h(s, a)$ known

Inside iteration n :

Use all previous data to estimate dynamics $\{\hat{P}_h^n\}_{h=0}^{H-1}$

Design reward bonus $b_h^n(s, a), \forall s, a, h$

UCB-VI: Tabular optimism in the face of uncertainty

Assume reward function $r_h(s, a)$ known

Inside iteration n :

Use all previous data to estimate dynamics $\{\hat{P}_h^n\}_{h=0}^{H-1}$

Design reward bonus $b_h^n(s, a), \forall s, a, h$

Optimistic planning with learned model: $\pi^n = \text{VI}\left(\{\hat{P}_h^n, r_h + b_h^n\}_{h=1}^{H-1}\right)$

UCB-VI: Tabular optimism in the face of uncertainty

Assume reward function $r_h(s, a)$ known

Inside iteration n :

Use all previous data to estimate dynamics $\{\hat{P}_h^n\}_{h=0}^{H-1}$

Design reward bonus $b_h^n(s, a), \forall s, a, h$

Optimistic planning with learned model: $\pi^n = \text{VI}\left(\{\hat{P}_h^n, r_h + b_h^n\}_{h=1}^{H-1}\right)$

Collect a new trajectory by executing π^n in the true system $\{P_h\}_{h=0}^{H-1}$ starting from s_0

Model Estimation

Let us consider the **very beginning** of episode n:

$$\mathcal{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h$$

Model Estimation

Let us consider the **very beginning** of episode n:

$$\mathcal{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h$$

Let's also maintain some statistics using these datasets:

Model Estimation

Let us consider the **very beginning** of episode n:

$$\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h$$

Let's also maintain some statistics using these datasets:

$$N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \quad \forall s, a, h,$$

Model Estimation

Let us consider the **very beginning** of episode n:

$$\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h$$

Let's also maintain some statistics using these datasets:

$$N_h^n(s, a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s, a)\}, \quad \forall s, a, h,$$

$$N_h^n(s, a, s') = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i, s_{h+1}^i) = (s, a, s')\}, \quad \forall s, a, s', h$$

Model Estimation

Let us consider the **very beginning** of episode n:

$$\mathcal{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h$$

Let's also maintain some statistics using these datasets:

$$N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \quad \forall s, a, h,$$

$$N_h^n(s,a,s') = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i, s_{h+1}^i) = (s,a,s')\}, \quad \forall s, a, s', h$$

Estimate model $\hat{P}_h^n(s' \mid s,a), \forall s, a, s', h$:

$$\hat{P}_h^n(s' \mid s,a) = \frac{N_h^n(s,a,s')}{N_h^n(s,a)}$$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Reward Bonus Design and Value Iteration

Recall: $\mathcal{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

$\hat{V}_H^n(s) = 0, \quad \forall s$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

$$\hat{V}_H^n(s) = 0, \ \forall s \qquad \hat{Q}_h^n(s,a) = \min\left\{ r_h(s,a) + b_h^n(s,a) + \mathbb{E}_{s' \sim \hat{P}_h^n(\cdot|s,a)}\left[\hat{V}_{h+1}^n(s')\right], \quad H \right\}, \ \forall s, a$$

Reward Bonus Design and Value Iteration

Recall: $\mathcal{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\frac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

$$\hat{V}_H^n(s) = 0, \ \forall s \qquad \hat{Q}_h^n(s,a) = \min\left\{ r_h(s,a) + b_h^n(s,a) + \mathbb{E}_{s' \sim \hat{P}_h^n(\cdot|s,a)}\left[\hat{V}_{h+1}^n(s')\right], \quad H \right\}, \ \forall s, a$$

$$\hat{V}_h^n(s) = \max_a \hat{Q}_h^n(s,a), \quad \pi_h^n(s) = \arg\max_a \hat{Q}_h^n(s,a), \ \forall s$$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\dfrac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

$$\hat{V}_H^n(s) = 0, \ \forall s \qquad \hat{Q}_h^n(s,a) = \min\left\{ r_h(s,a) + b_h^n(s,a) + \mathbb{E}_{s'\sim \hat{P}_h^n(\cdot|s,a)}\left[\hat{V}_{h+1}^n(s')\right], \quad H \right\}, \ \forall s, a$$

$$\hat{V}_h^n(s) = \max_a \hat{Q}_h^n(s,a), \quad \pi_h^n(s) = \arg\max_a \hat{Q}_h^n(s,a), \ \forall s$$

$$\left\| \hat{V}_h^n \right\|_\infty \leq H, \ \forall h, n$$

Reward Bonus Design and Value Iteration

Recall: $\mathscr{D}_h^n = \{s_h^i, a_h^i, s_{h+1}^i\}_{i=1}^{n-1}, \forall h, \quad N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h,$

Define: $b_h^n(s,a) = cH\sqrt{\dfrac{\log\left(|S||A|HN/\delta\right)}{N_h^n(s,a)}}$

Encourage to explore new state-actions

Value Iteration (aka DP) at episode n using $\{\hat{P}_h^n\}_h$ and $\{r_h + b_h^n\}_h$

$$\hat{V}_H^n(s) = 0, \ \forall s \qquad \hat{Q}_h^n(s,a) = \min\left\{ r_h(s,a) + b_h^n(s,a) + \mathbb{E}_{s' \sim \hat{P}_h^n(\cdot|s,a)}\left[\hat{V}_{h+1}^n(s')\right], \quad H \right\}, \ \forall s, a$$

$$\hat{V}_h^n(s) = \max_a \hat{Q}_h^n(s,a), \quad \pi_h^n(s) = \arg\max_a \hat{Q}_h^n(s,a), \ \forall s$$

$$\left\| \hat{V}_h^n \right\|_\infty \leq H, \ \forall h, n$$

$b_h^n(s,a)$ specifically chosen so that $V_h^\star(s) \leq \hat{V}_h^n(s)$ with high probability

UCBVI: Put All Together

For $n = 1 \to N$:

1. Set $N_h^n(s,a) = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i) = (s,a)\}, \forall s, a, h$

2. Set $N_h^n(s,a,s') = \sum_{i=1}^{n-1} \mathbf{1}\{(s_h^i, a_h^i, s_{h+1}^i) = (s,a,s')\}, \forall s, a, a', h$

3. Estimate $\hat{P}^n$: $\hat{P}_h^n(s' \mid s,a) = \dfrac{N_h^n(s,a,s')}{N_h^n(s,a)}, \forall s, a, s', h$

4. Plan: π^n = VI $\left(\{\hat{P}_h^n, r_h + b_h^n\}_h \right)$, with $b_h^n(s,a) = cH\sqrt{\dfrac{\log(|S||A|HN/\delta)}{N_h^n(s,a)}}$

5. Execute π^n : $\{s_0^n, a_0^n, r_0^n, \ldots, s_{H-1}^n, a_{H-1}^n, r_{H-1}^n, s_H^n\}$

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

Then π^n is close to $\pi^\star$, i.e., we are doing exploitation

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

Then π^n is close to $\pi^\star$, i.e., we are doing exploitation

2. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is large?

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

Then π^n is close to $\pi^\star$, i.e., we are doing exploitation

2. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is large?

Some $b_h^n(s,a)$ must be large (or some $\hat{P}_h^n(\cdot \mid s,a)$ estimation errors must be large, but with high probability any $\hat{P}_h^n(\cdot \mid s,a)$ with high error must have small $N_h^n(s,a)$ and hence high $b_h^n(s,a)$)

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

Then π^n is close to $\pi^\star$, i.e., we are doing exploitation

2. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is large?

Some $b_h^n(s,a)$ must be large (or some $\hat{P}_h^n(\cdot \mid s,a)$ estimation errors must be large, but with high probability any $\hat{P}_h^n(\cdot \mid s,a)$ with high error must have small $N_h^n(s,a)$ and hence high $b_h^n(s,a)$)

Large $b_h^n(s,a)$ means π^n is being encouraged to do (s,a), since it will apparently have very high reward, i.e., exploration

High-level Idea: Exploration Exploitation Tradeoff

Upper bound per-episode regret: $V_0^\star(s_0) - V_0^{\pi^n}(s_0) \leq \hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ by construction of b_h^n

1. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is small?

Then π^n is close to $\pi^\star$, i.e., we are doing exploitation

2. What if $\hat{V}_0^n(s_0) - V_0^{\pi^n}(s_0)$ is large?

Some $b_h^n(s,a)$ must be large (or some $\hat{P}_h^n(\cdot \mid s,a)$ estimation errors must be large, but with high probability any $\hat{P}_h^n(\cdot \mid s,a)$ with high error must have small $N_h^n(s,a)$ and hence high $b_h^n(s,a)$)

Large $b_h^n(s,a)$ means π^n is being encouraged to do (s,a), since it will apparently have very high reward, i.e., exploration

$$\mathbb{E}\left[\text{Regret}_N\right] := \mathbb{E}\left[\sum_{n=1}^{N}\left(V^\star - V^{\pi^n}\right)\right] \leq \widetilde{O}\left(H^2\sqrt{|S||A|N}\right)$$

Today

- ✓ Feedback from last lecture
- ✓ Recap
- ✓ Warm-up: ExploreThenExploit for deterministic MDPs
- ✓ Why we don't want to treat MDPs as big bandits
- ✓ UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

Linear MDP Definition

Finite horizon time-dependent episodic MDP $\mathscr{M} = \{S, A, H, \{r\}_h, \{P\}_h, s_0\}$

S & A could be large or even continuous, hence poly$(|S|, |A|)$ is not acceptable

Linear MDP Definition

Finite horizon time-dependent episodic MDP $\mathscr{M} = \{S, A, H, \{r\}_h, \{P\}_h, s_0\}$

S & A could be large or even continuous, hence poly$(|S|, |A|)$ is not acceptable

$$P_h(s'|s,a) = \mu_h^\star(s') \cdot \phi(s,a), \quad \mu_h^\star : S \mapsto \mathbb{R}^d, \quad \phi : S \times A \mapsto \mathbb{R}^d$$

Linear MDP Definition

Finite horizon time-dependent episodic MDP $\mathscr{M} = \{S, A, H, \{r\}_h, \{P\}_h, s_0\}$

S & A could be large or even continuous, hence poly$(|S|, |A|)$ is not acceptable

$$P_h(s' | s, a) = \mu_h^\star(s') \cdot \phi(s, a), \quad \mu_h^\star : S \mapsto \mathbb{R}^d, \quad \phi : S \times A \mapsto \mathbb{R}^d$$

$$r(s, a) = \theta_h^\star \cdot \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

Linear MDP Definition

Finite horizon time-dependent episodic MDP $\mathscr{M} = \{S, A, H, \{r\}_h, \{P\}_h, s_0\}$

S & A could be large or even continuous, hence poly$(|S|, |A|)$ is not acceptable

$$P_h(s'|s,a) = \mu_h^\star(s') \cdot \phi(s,a), \quad \mu_h^\star : S \mapsto \mathbb{R}^d, \quad \phi : S \times A \mapsto \mathbb{R}^d$$

$$r(s,a) = \theta_h^\star \cdot \phi(s,a), \quad \theta_h^\star \in \mathbb{R}^d$$

Feature map ϕ is known to the learner!

(We assume reward is known, i.e., $\theta^\star$ is known)

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \;\; \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \; \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$V_H^\star(s) = 0, \forall s,$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \;\; \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$V_H^\star(s) = 0, \forall s,$

$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \ \ \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$$V_H^\star(s) = 0, \forall s,$$

$$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$$

$$= \theta_h^\star \cdot \phi(s, a) + \left(\mu_h^\star \phi(s, a)\right)^\top V_{h+1}^\star$$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \;\; \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$$V_H^\star(s) = 0, \forall s,$$

$$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$$

$$= \theta_h^\star \cdot \phi(s, a) + \left(\mu_h^\star \phi(s, a)\right)^\top V_{h+1}^\star$$

$$= \phi(s, a)^\top \left(\theta_h^\star + (\mu_h^\star)^\top V_{h+1}^\star\right)$$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \;\; \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$$V_H^\star(s) = 0, \forall s,$$

$$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$$

$$= \theta_h^\star \cdot \phi(s, a) + \left(\mu_h^\star \phi(s, a)\right)^\top V_{h+1}^\star$$

$$= \phi(s, a)^\top \left(\theta_h^\star + (\mu_h^\star)^\top V_{h+1}^\star\right)$$

$$= \phi(s, a)^\top w_h$$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S|\times d}, \ \ \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$$V_H^\star(s) = 0, \forall s,$$

$$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$$

$$= \theta_h^\star \cdot \phi(s, a) + \left(\mu_h^\star \phi(s, a)\right)^\top V_{h+1}^\star$$

$$= \phi(s, a)^\top \left(\theta_h^\star + (\mu_h^\star)^\top V_{h+1}^\star\right)$$

$$= \phi(s, a)^\top w_h$$

$$V_h^\star(s) = \max_a \phi(s, a)^\top w_h, \quad \pi_h^\star(s) = \arg\max_a \phi(s, a)^\top w_h$$

Planning in Linear MDP: Value Iteration

$$P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a), \quad \mu_h^\star \in \mathbb{R}^{|S| \times d}, \ \ \phi(s, a) \in \mathbb{R}^d$$

$$r_h(s, a) = (\theta_h^\star)^\top \phi(s, a), \quad \theta_h^\star \in \mathbb{R}^d$$

$$V_H^\star(s) = 0, \forall s,$$

$$Q_h^\star(s, a) = r_h(s, a) + \mathbb{E}_{s' \sim P_h(\cdot|s,a)} V_{h+1}^\star(s')$$

$$= \theta_h^\star \cdot \phi(s, a) + \left(\mu_h^\star \phi(s, a)\right)^\top V_{h+1}^\star$$

$$= \phi(s, a)^\top \left(\theta_h^\star + (\mu_h^\star)^\top V_{h+1}^\star\right)$$

$$= \phi(s, a)^\top w_h$$

$$V_h^\star(s) = \max_a \phi(s, a)^\top w_h, \quad \pi_h^\star(s) = \arg\max_a \phi(s, a)^\top w_h$$

Indeed we can show that $Q_h^\pi(\cdot, \cdot)$
Is linear with respect to ϕ as well, for any π, h

UCBVI in Linear MDPs

At the beginning of iteration n:

UCBVI in Linear MDPs

At the beginning of iteration n:

1. Learn transition model $\{\hat{P}_h^n\}_{h=0}^{H-1}$ from all previous data $\{s_h^i, a_h^i, s_{h+1}^i\}_{i=0}^{n-1}$

UCBVI in Linear MDPs

At the beginning of iteration n:

1. Learn transition model $\{\hat{P}_h^n\}_{h=0}^{H-1}$ from all previous data $\{s_h^i, a_h^i, s_{h+1}^i\}_{i=0}^{n-1}$

2. Design reward bonus $b_h^n(s, a), \forall s, a$

UCBVI in Linear MDPs

At the beginning of iteration n:

1. Learn transition model $\{\hat{P}_h^n\}_{h=0}^{H-1}$ from all previous data $\{s_h^i, a_h^i, s_{h+1}^i\}_{i=0}^{n-1}$

2. Design reward bonus $b_h^n(s, a), \forall s, a$

3. Plan: $\pi^{n+1} = \text{VI}\left(\{\hat{P}^n\}_h, \{r_h + b_h^n\} \right)$

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

Denote $\delta(s) \in \mathbb{R}^{|S|}$ with zero everywhere except the entry corresponding to s

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

Denote $\delta(s) \in \mathbb{R}^{|S|}$ with zero everywhere except the entry corresponding to s

Given s, a, note that $\mathbb{E}_{s' \sim P_h(\cdot|s,a)}\left[\delta(s')\right] = P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a)$

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

Denote $\delta(s) \in \mathbb{R}^{|S|}$ with zero everywhere except the entry corresponding to s

Given s, a, note that $\mathbb{E}_{s' \sim P_h(\cdot|s,a)} \left[\delta(s')\right] = P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a)$

Penalized Linear Regression:

$$\min_{\mu} \sum_{i=1}^{n-1} \|\mu\phi(s_h^i, a_h^i) - \delta(s_{h+1}^i)\|_2^2 + \lambda\|\mu\|_F^2$$

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

Denote $\delta(s) \in \mathbb{R}^{|S|}$ with zero everywhere except the entry corresponding to s

Given s, a, note that $\mathbb{E}_{s' \sim P_h(\cdot|s,a)}\left[\delta(s')\right] = P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a)$

Penalized Linear Regression:

$$\min_{\mu} \sum_{i=1}^{n-1} \|\mu\phi(s_h^i, a_h^i) - \delta(s_{h+1}^i)\|_2^2 + \lambda\|\mu\|_F^2$$

$$A_h^n = \sum_{i=1}^{n-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\top + \lambda I$$

$$\hat{\mu}_h^n = (A_h^n)^{-1} \sum_{i=1}^{n-1} \delta(s_{h+1}^i)\phi(s_h^i, a_h^i)^\top$$

How to estimate $\{\hat{P}_h^n\}_{h=0}^{H-1}$?

Denote $\delta(s) \in \mathbb{R}^{|S|}$ with zero everywhere except the entry corresponding to s

Given s, a, note that $\mathbb{E}_{s' \sim P_h(\cdot|s,a)}\left[\delta(s')\right] = P_h(\cdot \mid s, a) = \mu_h^\star \phi(s, a)$

Penalized Linear Regression:

$$\min_{\mu} \sum_{i=1}^{n-1} \|\mu \phi(s_h^i, a_h^i) - \delta(s_{h+1}^i)\|_2^2 + \lambda \|\mu\|_F^2$$

$$A_h^n = \sum_{i=1}^{n-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\top + \lambda I$$

$$\hat{\mu}_h^n = (A_h^n)^{-1} \sum_{i=1}^{n-1} \delta(s_{h+1}^i)\phi(s_h^i, a_h^i)^\top$$

$$\hat{P}_h^n(\cdot \mid s, a) = \hat{\mu}_h^n \phi(s, a)$$

How to choose $b_h^n(s,a)$?

Chebyshev-like approach, similar to in linUCB (will cover next lecture):

$$b_h^n(s,a) = \beta\sqrt{\phi(s,a)^\top (A_h^n)^{-1}\phi(s,a)}, \quad \beta = \widetilde{O}(dH)$$

linUCB-VI: Put All Together

For $n = 1 \to N$:

1. Set $A_h^n = \sum_{i=1}^{n-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\top + \lambda I$

2. Set $\widehat{\mu}_h^n = (A_h^n)^{-1} \sum_{i=1}^{n-1} \delta(s_{h+1}^i)\phi(s_h^i, a_h^i)^\top$

3. Estimate $\hat{P}^n : \hat{P}_h^n(\cdot \mid s, a) = \widehat{\mu}_h^n \phi(s, a)$

4. Plan: $\pi^n = \text{VI}\left(\{\hat{P}_h^n, r_h + b_h^n\}_h\right)$, with $b_h^n(s, a) = cdH\sqrt{\phi(s, a)^\top (A_h^n)^{-1} \phi(s, a)}$

5. Execute $\pi^n : \{s_0^n, a_0^n, r_0^n, \ldots, s_{H-1}^n, a_{H-1}^n, r_{H-1}^n, s_H^n\}$

linUCB-VI: Put All Together

For $n = 1 \to N$:

1. Set $A_h^n = \sum_{i=1}^{n-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\top + \lambda I$
2. Set $\widehat{\mu}_h^n = (A_h^n)^{-1} \sum_{i=1}^{n-1} \delta(s_{h+1}^i)\phi(s_h^i, a_h^i)^\top$
3. Estimate $\hat{P}^n$: $\hat{P}_h^n(\cdot \mid s, a) = \widehat{\mu}_h^n \phi(s, a)$
4. Plan: $\pi^n = \text{VI}\left(\{\hat{P}_h^n, r_h + b_h^n\}_h\right)$, with $b_h^n(s, a) = cdH\sqrt{\phi(s, a)^\top (A_h^n)^{-1} \phi(s, a)}$
5. Execute π^n : $\{s_0^n, a_0^n, r_0^n, \ldots, s_{H-1}^n, a_{H-1}^n, r_{H-1}^n, s_H^n\}$

$$\mathbb{E}\left[\text{Regret}_N\right] := \mathbb{E}\left[\sum_{n=1}^{N}\left(V^\star - V^{\pi^n}\right)\right] \leq \widetilde{O}\left(H^2 d^{1.5}\sqrt{N}\right)$$

linUCB-VI: Put All Together

For $n = 1 \to N$:

1. Set $A_h^n = \sum_{i=1}^{n-1} \phi(s_h^i, a_h^i)\phi(s_h^i, a_h^i)^\top + \lambda I$

2. Set $\widehat{\mu}_h^n = (A_h^n)^{-1} \sum_{i=1}^{n-1} \delta(s_{h+1}^i)\phi(s_h^i, a_h^i)^\top$

3. Estimate $\hat{P}^n : \hat{P}_h^n(\cdot \mid s, a) = \widehat{\mu}_h^n \phi(s, a)$

4. Plan: $\pi^n = \text{VI}\left(\{\hat{P}_h^n, r_h + b_h^n\}_h\right)$, with $b_h^n(s, a) = cdH\sqrt{\phi(s, a)^\top (A_h^n)^{-1} \phi(s, a)}$

5. Execute $\pi^n : \{s_0^n, a_0^n, r_0^n, \ldots, s_{H-1}^n, a_{H-1}^n, r_{H-1}^n, s_H^n\}$

$$\mathbb{E}\left[\text{Regret}_N\right] := \mathbb{E}\left[\sum_{n=1}^{N} \left(V^\star - V^{\pi^n}\right)\right] \leq \widetilde{O}\left(H^2 d^{1.5}\sqrt{N}\right)$$

No S, A dependence!

Today

- Feedback from last lecture
- Recap
- Warm-up: ExploreThenExploit for deterministic MDPs
- Why we don't want to treat MDPs as big bandits
- UCB-VI for tabular MDPs
- UCB-VI for linear MDPs

Summary:

UCBVI algorithm applies UCB idea to MDPs to achieve exploration/exploitation trade-off

Attendance:

bit.ly/3RcTC9T

Feedback:

bit.ly/3RHtlxy